Neural Networks & Deep Learning

ICP-3

Sreeja Reddy Konda

700756597

GitHub-

https://github.com/SreejaReddyKonda/Neural-Network-Sreeja/blob/main/Neural%20Networks/ICP-4/ICP-4.ipynb

Video-

https://drive.google.com/file/d/11XOCycMxNzGOmX7N5fahwyZbd6yM25NH/view?usp=drive_link

1.

```
[12] # existing code
     import pandas
     from keras.models import Sequential
     from keras.layers import Dense, Activation

     # load dataset
     from sklearn.model_selection import train_test_split
     import pandas as pd
     import numpy as np

     dataset = pd.read_csv("diabetes.csv", header=None).values

     X_train, X_test, Y_train, Y_test = train_test_split(dataset[:,0:8], dataset[:,8],
                                                         test_size=0.25, random_state=87)
     np.random.seed(155)
     my_first_nn = Sequential() # create model
     my_first_nn.add(Dense(20, input_dim=8, activation='relu')) # hidden layer
     my_first_nn.add(Dense(1, activation='sigmoid')) # output layer
     my_first_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
     my_first_nn_fitted = my_first_nn.fit(X_train, Y_train, epochs=100,
                                          initial_epoch=0)
     print(my_first_nn.summary())
     print(my_first_nn.evaluate(X_test, Y_test))
```

```
Epoch 1/100
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
18/18 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - acc: 0.3215 - loss: 13.4017
Epoch 2/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.4512 - loss: 6.0000
Epoch 3/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.5977 - loss: 2.3603
Epoch 4/100
```

```
:poch 100/100
18/18 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6997 - loss: 0.6185
lodel: "sequential_14"
```

Layer (type)	Output Shape	Param #
dense_38 (Dense)	(None, 20)	180
dense_39 (Dense)	(None, 1)	21

```
Total params: 605 (2.37 KB)
Trainable params: 201 (804.00 B)
Non-trainable params: 0 (0.00 B)
Optimizer params: 404 (1.58 KB)
lone
5/6 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.6862 - loss: 0.6685
[0.6848695278167725, 0.6770833134651184]
```

[2]
```
# Add more Dense layers to the existing code and check how the accuracy changes


from google.colab import drive
drive.mount('/content/gdrive')

path_to_csv = '/content/gdrive/My Drive/breastcancer.csv'

import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

# loading dataset
cancer_data = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_nn_model = Sequential() # creating the model
my_nn_model.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
my_nn_model.add(Dense(32, activation='relu'))
my_nn_model.add(Dense(1, activation='sigmoid')) # output layer
my_nn_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn_model.fit(X_train, Y_train, epochs=100,
                           initial_epoch=0)
print(my_nn_model.summary())
print(my_nn_model.evaluate(X_test, Y_test))
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.9353 - loss: 0.2041
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 20)	620
dense_5 (Dense)	(None, 32)	672
dense_6 (Dense)	(None, 1)	33

```
 Total params: 3,977 (15.54 KB)
 Trainable params: 1,325 (5.18 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2,652 (10.36 KB)
None
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - acc: 0.8361 - loss: 0.4254
[0.35378405451774597, 0.8741258978843689]
```

[4]
```python
# Normalize the data before feeding the data to the model and check how the normalization change your accuracy
from google.colab import drive
drive.mount('/content/gdrive')

path_to_csv = '/content/gdrive/My Drive/breastcancer.csv'

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

import keras
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

# loading dataset
cancer_data = load_breast_cancer()
X_train, X_test, Y_train, Y_test = train_test_split(cancer_data.data, cancer_data.target,
                                                    test_size=0.25, random_state=87)
np.random.seed(155)
my_nn_model = Sequential() # creating model
my_nn_model.add(Dense(20, input_dim=30, activation='relu')) # hidden layer 1
my_nn_model.add(Dense(1, activation='sigmoid')) # output layer
my_nn_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
my_nn_fitted = my_nn_model.fit(X_train, Y_train, epochs=100,
                               initial_epoch=0)
print(my_nn_model.summary())
print(my_nn_model.evaluate(X_test, Y_test))
```

14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - acc: 0.9525 - loss: 0.2215

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dense_7 (Dense)	(None, 20)	620
dense_8 (Dense)	(None, 1)	21

Total params: 1,925 (7.52 KB)
Trainable params: 641 (2.50 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 1,284 (5.02 KB)

None

5/5 ━━━━━━━━━━━━━━━━━━━━ **0s** 3ms/step - acc: 0.8698 - loss: 0.4206

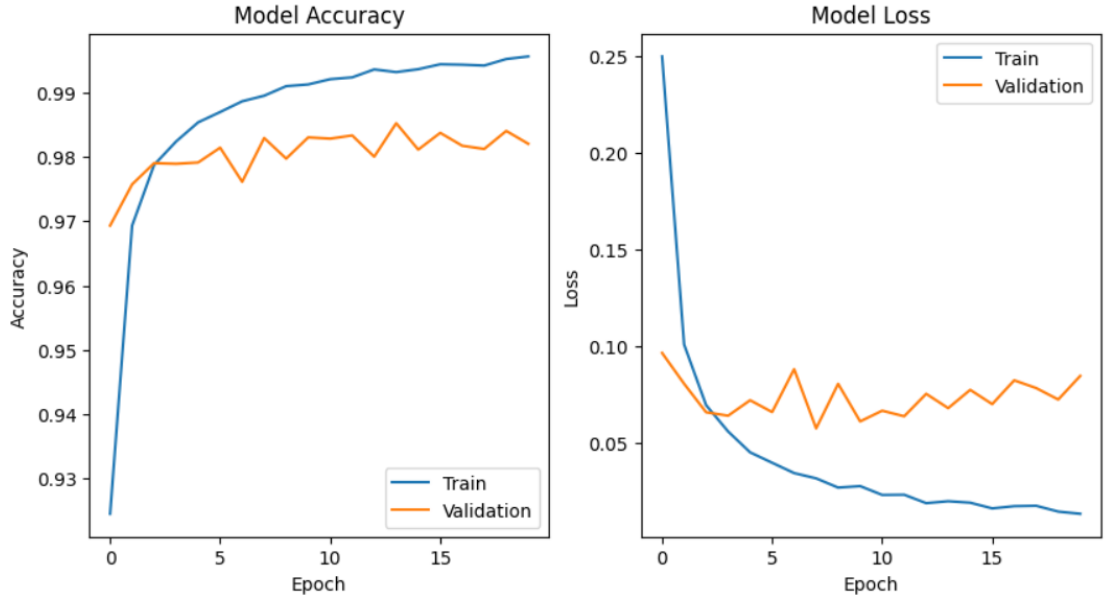
[0.33301880955696106, 0.8881118893623352]

2.

[6]
```python
# Plot the loss and accuracy for both training data and validation data using the history object in the source code
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt

# load mnist dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# converting class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# creating a simple neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# training the model and record the training history
history = model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
                    epochs=20, batch_size=128)
```

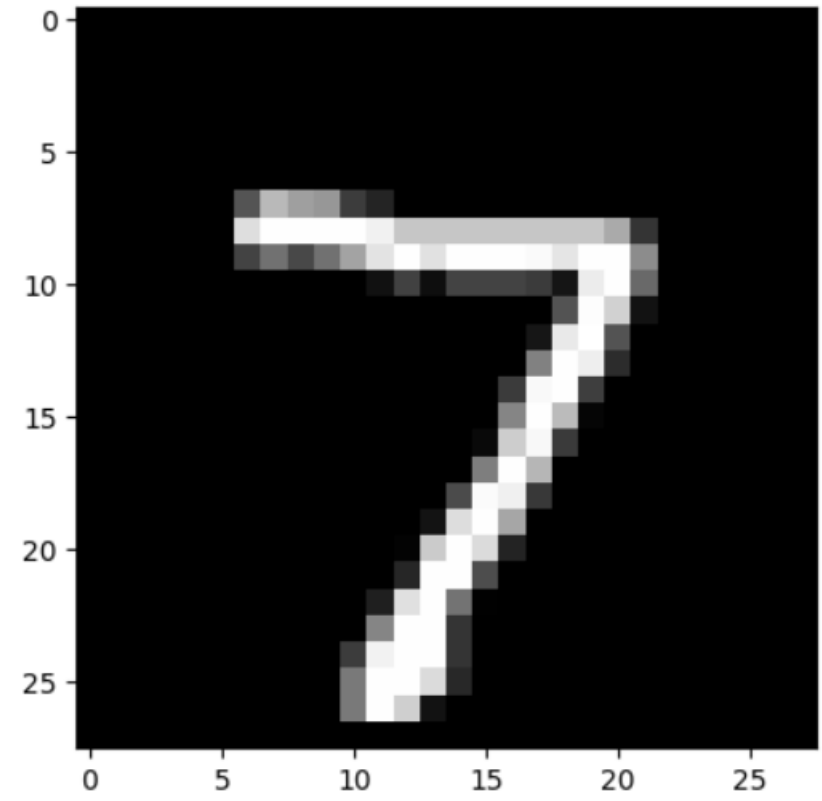
```python
# to plot the training and validation accuracy and loss curves
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='lower right')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper right')
```

```
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - accuracy: 0.9958 - loss: 0.0125 - val_accuracy: 0.9820 - val_loss: 0.0846
```



[7]
```
#  Plot one of the images in the test data, and then do inferencing to check what is the prediction of the model on that single image.
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# loading mnist dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# converting class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# to create a simple neural network model
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# to train the model
model.fit(x_train.reshape(-1, 784), y_train, validation_data=(x_test.reshape(-1, 784), y_test),
          epochs=20, batch_size=128)
```

```
# to plot one of the images in the test data
plt.imshow(x_test[0], cmap='gray')
plt.show()

# making a prediction on the image using the trained model
prediction = model.predict(x_test[0].reshape(1, -1))
print('Model prediction:', np.argmax(prediction))
```

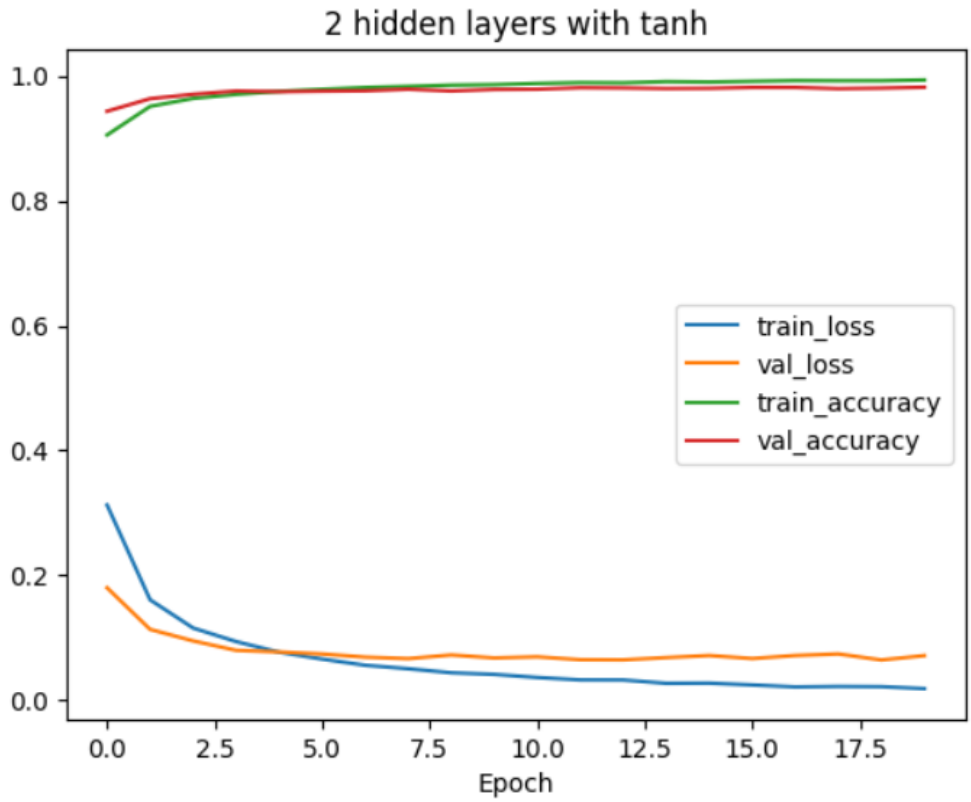



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 101ms/step
Model prediction: 7
```

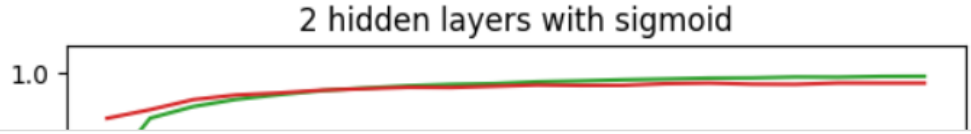
```python
# We had used 2 hidden layers and Relu activation. Try to change the number of hidden layer and the activation to
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# normalize pixel values to range [0, 1]
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a list of models to train
models = []

# model with 1 hidden layer and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
```

[8]

```
1 hidden layer with sigmoid - Test loss: 0.0619, Test accuracy: 0.9814
```



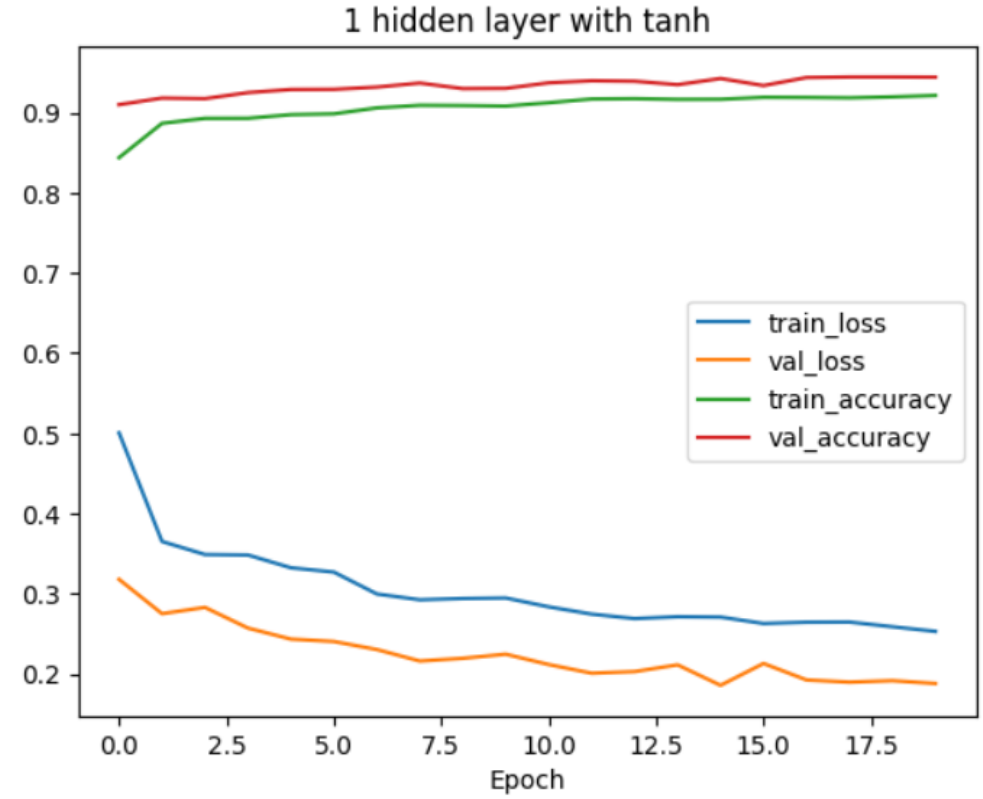

```
2 hidden layers with tanh - Test loss: 0.0713, Test accuracy: 0.9820
```



✓ 10s completed at 6:39 PM

[9]
```
# Run the same code without scaling the images and check the performance
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np

# load mnist dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# convert class labels to binary class matrices
num_classes = 10
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

# create a list of models to train
models = []

# model with 1 hidden layer and tanh activation
model = Sequential()
model.add(Dense(512, activation='tanh', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with tanh', model))

# model with 1 hidden layer and sigmoid activation
model = Sequential()
model.add(Dense(512, activation='sigmoid', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
models.append(('1 hidden layer with sigmoid', model))

# model with 2 hidden layers and tanh activation
```

✓ 11m [9]




```
1 hidden layer with tanh - Test loss: 0.1882, Test accuracy: 0.9442
```